SACLANTCEN
Technical Memorandum No. 179

SACLANT ASW
RESEARCH CENTRE

DIGITAL FILTERING TECHNIQUES FOR BROADBAND BEAMFORMING

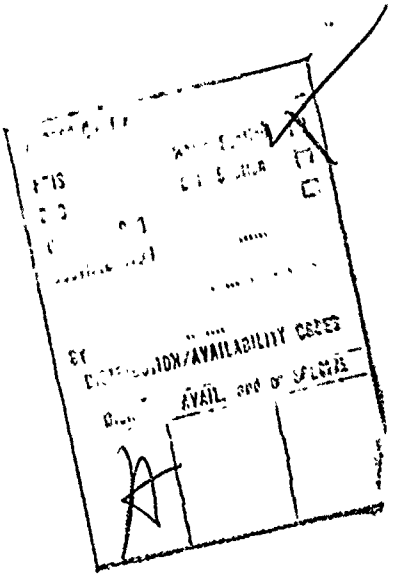
by

ARISTIDES A.G. REQUICHA

1 JULY 1972

NORTH
ATLANTIC
TREATY
ORGANIZATION

VIALE SAN BARTOLOMEO 400
I - 19026 - LA SPEZIA, ITALY



SACLANTCEN

TECHNICAL MEMORANDUM No. 179

NORTH ATLANTIC TREATY ORGANIZATION
SACLANT ASW RESEARCH CENTRE
Viale San Bartoloemo 400
I 19026 - La Spezia, Italy

DIGITAL FILTERING TECHNIQUES FOR BROADBAND BEAMFORMING

by

Aristides A.G. Requicha

1 July 1972

APPROVED FOR DISTRIBUTION

Ir M.W. van Batenburg
Director

TABLE OF CONTENTS

List of Figures

DIGITAL FILTERING TECHNIQUES FOR BROADBAND BEAMFORMING

by

Aristides A.G. Requicha

ABSTRACT

Digital steering of broadband hydrophone arrays is studied from the viewpoint of digital filtering theory. Emphasis is placed on applications wherein the sampling frequency must be kept as low as possible, implying that accurate beamforming involves interpolating the signals' samples. Finite-impulse-response filters are shown to provide convenient means for digital interpolation. Standard techniques for implementing such filters are reviewed and applied to the design of beamforming algorithms. Estimates are derived for the computing time and for the amount of core memory required by digital computer realizations of time-domain and frequency-domain beamforming processors. It is shown that frequency-domain processing may yield large savings in computing time when the sensors' outputs have to be filtered prior to steering; it can be competitive even for beamforming alone. This situation arises e.g. in "optimal" array processing, or in wideband arrays used in conjunction with explosive sound sources. It is pointed out that the digital filtering approach clarifies the approximations and possible pitfalls involved in the design and implementation of digital beamsteering processors.

INTRODUCTION

Electronically steered hydrophone arrays are used in underwater acoustics, especially when the transducer size or the scanning speed make mechanical steering unpractical. Because of the growing number of computers and of other digital equipment being used in the acquisition and processing of underwater acoustics data it is of interest to study digital techniques of array steering.

The work reported in this memorandum was a preliminary investigation on digital beamsteering techniques to be used in the design of beamformers for broadband arrays presently being built at SACLANTCEN.

The discussion is geared towards applications where the sampling frequency must be kept close to its minimum theoretical value for the data acquisition rate not to become unduly high. This situation is often encountered in experimental work involving hydrophone arrays and explosive sound sources. In these applications data acquisition rates are usually quite high and can only be increased at the cost of considerable equipment complexity.*

The digital filtering techniques described are equally applicable to hardware or software processor implementations. The discussion of computational details, however, is mostly pertinent to implementations in general purpose digital computers. The possibility of using a small shipborne computer to implement an array processor is very attractive, since the computer can be time-shared with other activities and effectively perform various data acquisition and analysis tasks at sea. Furthermore, software processors afford

---

* The maximum data acquisition rate of a system currently used at SACLANTCEN [Ref. 1] is 240 kHz, corresponding to a maximum number of 20 hydrophones at 12 kHz sampling frequency (or 10 at 24 kHz, etc.).

great flexibility and can be modified with relative ease. These considerations justify the emphasis on computer implementations. It should be noted, however, that hardware processors designed for a specific task can usually be made more efficient than a general purpose machine. An interesting compromise between flexibility and efficiency can be attained by extending the capabilities of a central processor with wired microprogrammed units, such as a fast Fourier transform (FFT) processor.

Beamforming consists essentially in delaying the hydrophone outputs by suitable amounts, followed by adding all the delayed signals. The beam is steered in a certain direction by introducing delays to compensate for the travel time differences between arrivals at the different hydrophones. For narrowband signals time-delaying can be implemented by means of phase-shifters, the usual solution in radar phased arrays. However, when the signals are wideband, as often happens in underwater acoustics, frequency-independent time delays (linear phase-shifts) must be introduced. This can be done using analog processing techniques by means of delay lines. Digital implementation is straightforward provided that the required delays are integral numbers of samples, and enough memory is available. Indeed, it suffices to store a matrix with a number of rows equal to the number of sensors, say, and a number of columns equal to the maximum delay needed, measured in samples. At each sampling instant, and for each beam, one need only "choose" from the matrix the elements with the correct delay and add them. Then, shift the data one column to the right, read in one new sample for each sensor, and so forth. Note that the beam can be steered in any direction by this technique within any prescribed accuracy merely by using a sampling frequency high enough for the delay quantization to have negligible effects.

When the sampling frequency is chosen fairly close to its theoretical minimum value, delays of fractions of a sample interval are usually needed. Consider, for example, a linear array with elements equally spaced half-a-wavelength apart at a frequency f. It is current practice at SACLANTCEN to sample at a frequency F_s equal to three times the maximum frequency of interest. A delay

of one sample (Δt) is equivalent to a phase shift of $2\pi f\Delta t$, i.e., 120° for $f=F_s/3$. In the example being discussed phase shifts of 90° between successive sensors are needed to steer the array at 30° from broadside (45° for 15° steering angle). Thus, time-delaying by integral number of samples is seen to lead to gross errors.

Digital beamforming techniques in the frequency domain, which allow the use of non-integral delays, have been described in recent letters to the J.A.S.A. [Refs. 2 and 3]. In the present paper digital beamforming is studied from the viewpoint of digital filtering theory. The full force of digital filtering techniques, developed over the past few years, can be called upon to design interpolating filters capable of approximating fractional delays to any prescribed accuracy. In this context, the FFT technique of Refs. 2 and 3 is recognized as a particular implementation by means of finite-impulse-response (FIR)* digital filters, and it becomes clear how to avoid the "wrap-around" errors mentioned in Ref. 3, which are due to the periodic nature of the discrete Fourier transform (DFT).

Digital filtering techniques for time-shifting a signal are discussed in Chapter 1. Because the problem of designing time-shifting (interpolating) filters is easier to solve for FIR filters, beamforming is discussed in this context in Chapter 2. Running time and memory requirements are estimated for time-domain and frequency-domain implementations of FIR beamformers. The time estimates given are rough and should be regarded as mere indications of the orders of magnitude involved. Digital beamforming in a computer is particularly attractive for broadband arrays such as those used in conjunction with explosive sound sources [Ref. 5]. For this type of applications the signals must be filtered prior to beamforming, and the computational effort for filtering and beamforming must be evaluated as a whole. This topic is included in the computational considerations of Chapter 2.

---

* Sometimes called "nonrecursive", although this terminology is somewhat incorrect [Ref. 4].

1. DIGITAL TIME-SHIFTING FILTERS

1.1 Shifting by an integral number of samples

Consider a sequence $\{x(n)\}$, obtained by sampling a continuous signal $x(t)$ at the time instants $\{n\Delta t\}$, where n is an integer. Time-shifting this sequence by $m\Delta t$, for integral m, is clearly equivalent to convolving the discrete signal with a Kronecker delta pulse located at sample number m:

$$x(n-m) = x(n)*\delta_{nm}$$

where $\delta_{nm}=1$ for $n=m$, and is zero otherwise. Direct implementation in the time domain is trivial; however, it is useful to point out that the convolution in the equation above can also be implemented (although inefficiently) in the frequency domain by using fast convolution techniques [Refs. 6 to 8]. Because of the cyclical nature of DFT-based convolution, care must be exercised in trying to implement aperiodic convolutions in the frequency domain. Standard procedures exist to handle this problem; the signal should be sectioned into blocks the length of which depends on the duration of the impulse response (IR) of the filters, and then particular techniques, such as "select-save" or "overlap-add", should be used. (The reader is referred to the above cited literature for details; an example using the "select-save" technique will be presented in section 1.3 below.)

1.2 Shifting by a fractional number of samples

Time-shifting a signal by a fractional number of samples consists essentially in interpolating the signal samples. It is most fruitful to design interpolating (time-shifting) filters starting from frequency-domain specifications, since it is usually in terms of frequency-domain tolerances that engineers "think". Furthermore, the design of digital filters to achieve specifications in the

frequency domain has been extensively studied, and numerous design techniques are available (see e.g. Refs. 4, 9, and 10).

The ideal frequency response of an interpolating digital filter is shown in Fig. 1. The slope of the linear phase characteristic is proportional to the time delay. (The periodic nature of the filter characteristics evident in Fig. 1 is of no consequence in applications since signals must be bandlimiter prior to being digitized.) For delays of a fraction of a sample the phase is discontinuous at the Nyquist frequency (defined as one-half the sampling frequency), implying that the imaginary component is also discontinuous. For this reason digital filter implementations will usually exhibit large errors in the neighbourhood of the Nyquist frequency. Note that no discontinuity exists for delays of an integral number of samples, because the phase is defined modulo 2π.

Designing a FIR digital filter to approximate characteristics of the type depicted in Fig. 1 is considerably simpler than designing an infinite-impulse-response (IIR) filter, and only FIR realizations will be discussed in this paper. Although rather sophisticated techniques exist for designing FIR filters [Ref. 4], a straightforward frequency-sampling procedure leads to accurate approximations with short IR's (low order filters), as the following example illustrates. Consider a half-a-sample delaying filter. Construct its ideal frequency response and sample it at M equidistant points. The IR of the filter is simply the inverse DFT (IDFT) of the frequency samples. M is the length of the IR measured in samples (order of the filter). The frequency response of the filter can be obtained from the frequency samples by trigonometric interpolation, which can be performed with the FFT [Ref. 11]. Comparison with the ideal characteristics yields the error in the approximation. The design method consists simply in evaluating the errors for various values of M, and in choosing the smallest M compatible with the tolerances. Magnitude and phase errors for the half-a-sample delay are shown in Fig. 2a for M=32, and in Fig. 2b for M=8. It is clear from the figures that high accuracy is obtained throughout most of the band with low values of M.

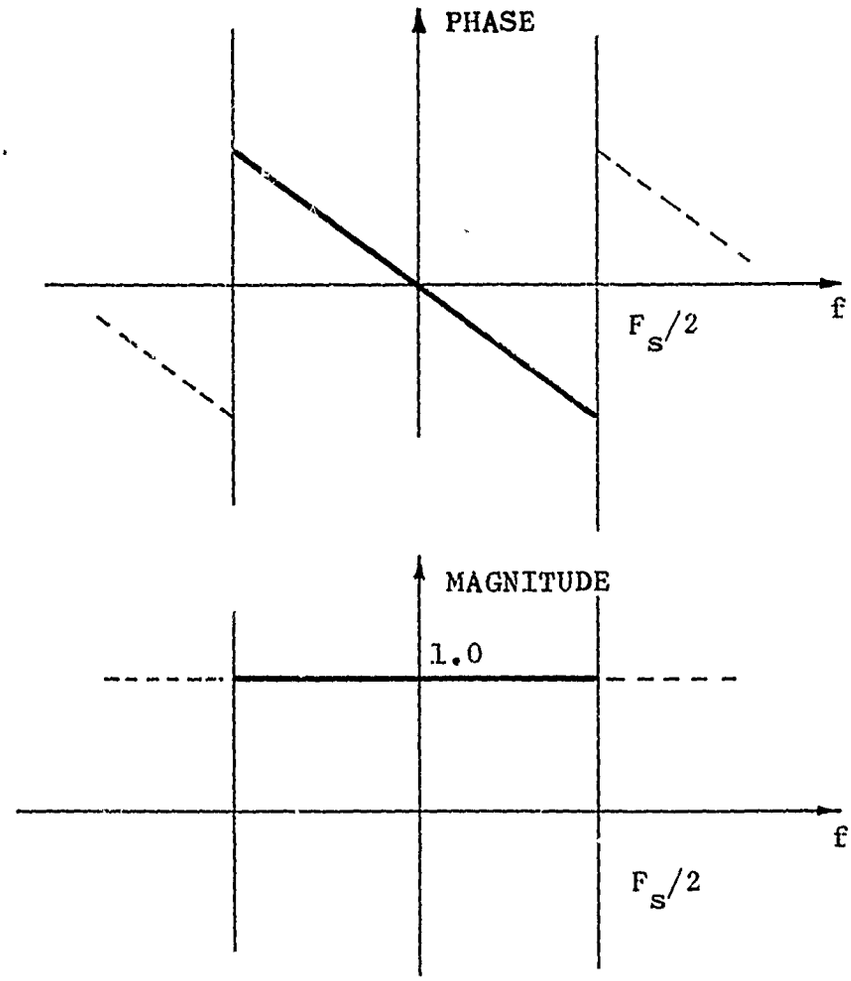


FIG. 1 PHASE AND AMPLITUDE CHARACTERISTICS OF AN IDEAL DIGITAL DELAYING FILTER

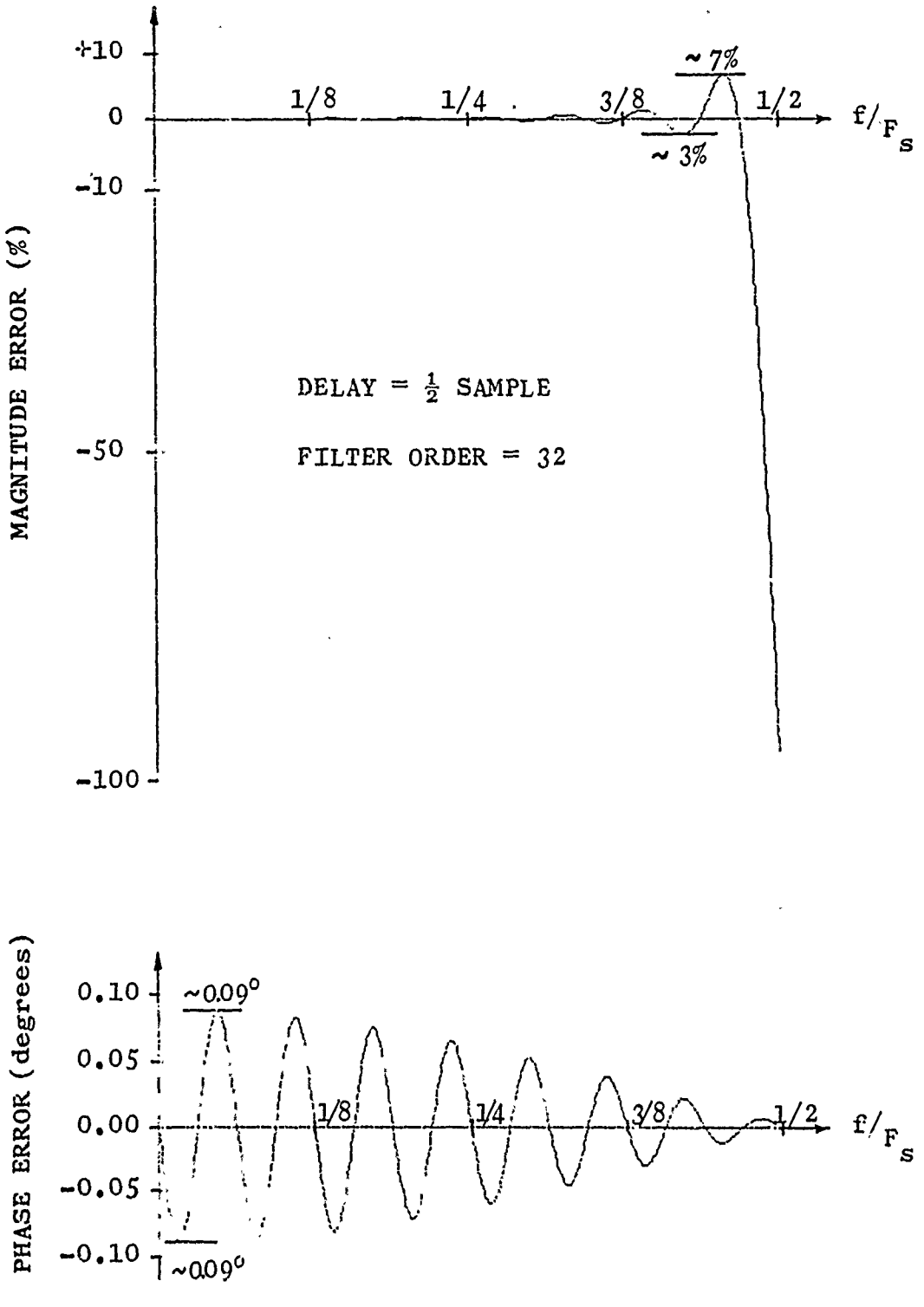


FIG. 2a MAGNITUDE AND PHASE ERRORS FOR ½ SAMPLE DELAYING FILTER OF ORDER 32

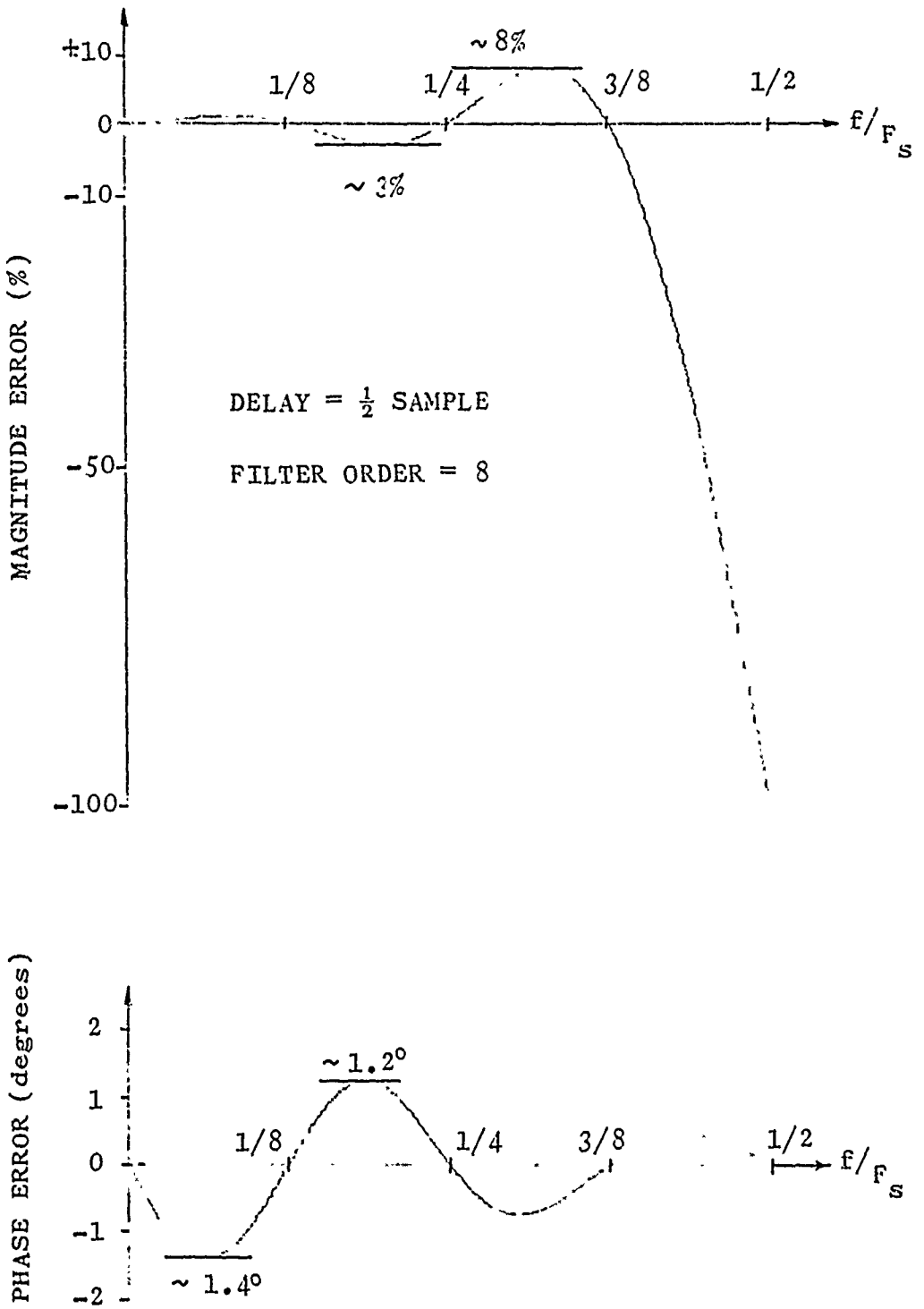


FIG. 2b MAGNITUDE AND PHASE ERRORS FOR ½ SAMPLE DELAYING FILTER OF ORDER 8

The most effective implementation of FIR shifting filters depends on the particular problem at hand. Nonrecursive realizations of FIR filters are usually more efficient than recursive realizations [Ref. 4]. There are two main choices of nonrecursive implementations: 1) direct convolution, which is generally to be preferred if the order of the filter is low and/or all the processing is done in the time domain, and 2) fast convolution using the FFT, advantageous if the IR is long and/or some other frequency-domain processing is needed.

1.3 An example of FFT-based implementation

To illustrate a procedure for implementing time-shifting filters using the FFT, a concrete example will be discussed in detail in this section.

Consider a delay of 1.5 samples. Note that it suffices to design a filter for $\frac{1}{2}$ sample delay and then to shift its IR by one sample to obtain a 1.5 samples delay. In the frequency domain this is equivalent to multiplying the DFT of the $\frac{1}{2}$ sample filter by $\{\exp(-j2\pi mn/N)\}$, where $m=1$, and N is the block size. Suppose that a 16th. order $\frac{1}{2}$ sample delaying filter designed by frequency sampling yields errors within the required tolerances. For an IR of length 16, the block length that leads to minimum computing time is 64 [Ref. 8]. The IR of a 1.5 sample delaying filter, shown in Fig. 3*, was obtained by first computing the DFT of the $\frac{1}{2}$ sample delay for a block size $N=64$, multiplying by $\{\exp(-j\pi n/8)\}$, and inverting.

For "anticipatory"** filters whose IR is zero for $n=L, L+1, \ldots, N-R$, the "select-save" technique described in Refs. 6 to 8 must be

---

*The solid lines connecting the samples are due to the linear interpolation used in the plotting routine.

**Strictly speaking the filter is not anticipatory, because FFT-processing automatically introduces a delay of one block size.

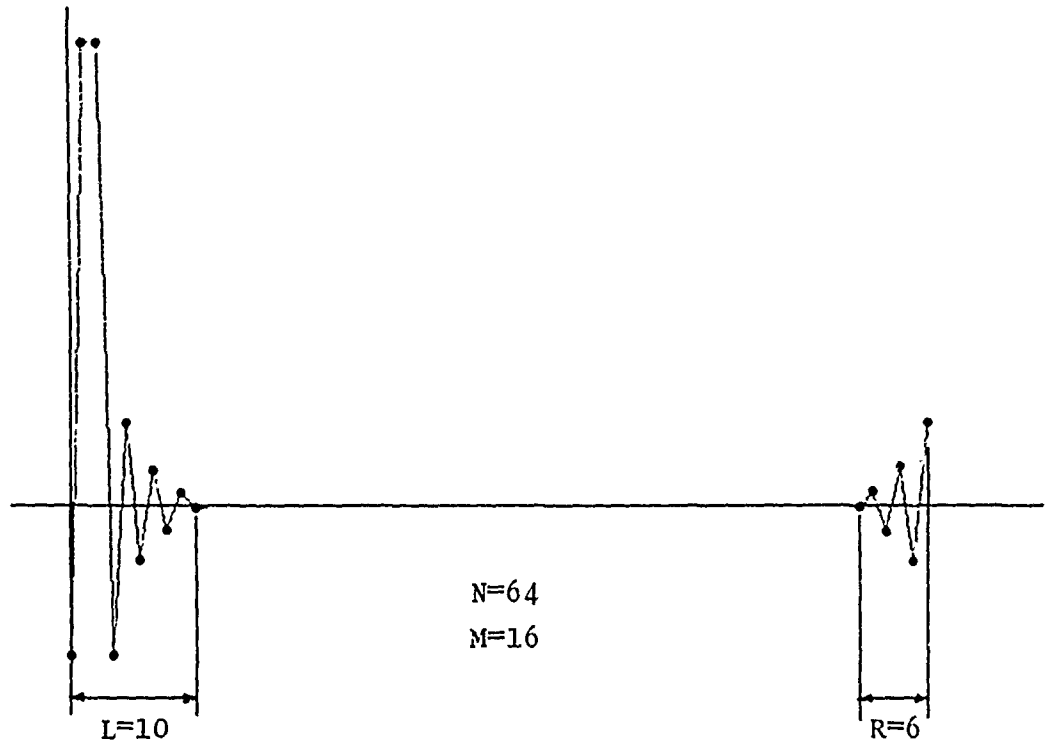


FIG. 3 IMPULSE RESPONSE OF 1½ SAMPLES DELAYING FILTER

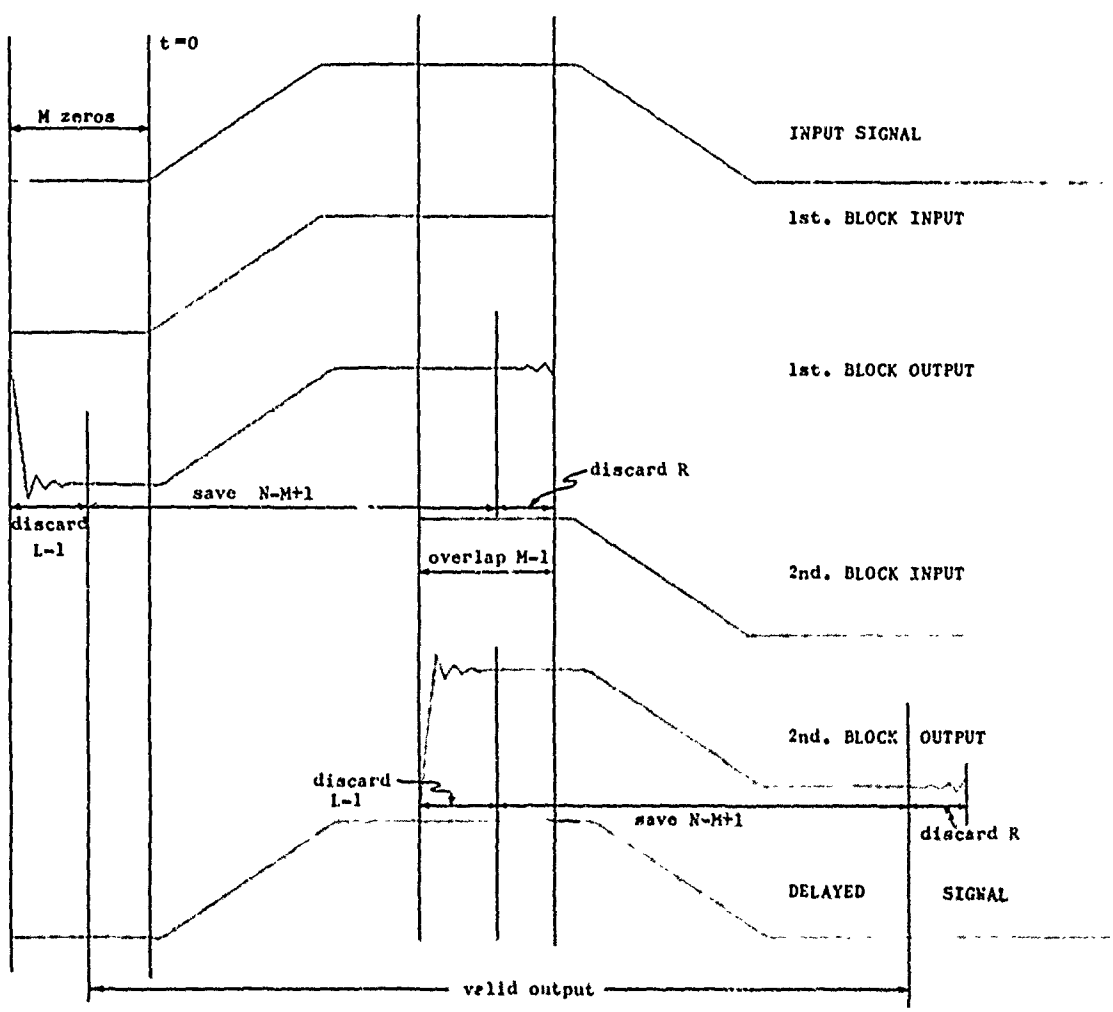


FIG. 4 ILLUSTRATION OF THE "SELECT-SAVE" TECHNIQUE

modified as indicated below. L and R are the effective lengths in samples of the "left" and "right" parts of the IR. For the example discussed here, L can be chosen equal to 10, and R to 6 (Fig. 3). Note, however, that L and R could be chosen as larger numbers and the procedure below would still be valid, although some efficiency would have been lost. Define the parameter M as the sum L+R. For the purposes of choosing the block size, it is this value of M that should be considered as the total length of the IR. The fast convolution can be implemented as follows:

1) Insert at least M-1 zeros before the first signal samples. (This is necessary to obtain the first M-1 samples of the result.) Construct with these zeros and the signal samples a first block of length N=64.

2) FFT the block.

3) Multiply by the DFT of the delaying filter.

4) IDFT and discard the first L-1 and the last R samples. The remaining N-M+1 samples are valid data.

5) Construct a new block of N points such that its first M-1 samples are the last M-1 samples of the previous block.

6) FFT, multiply, invert, discard, and continue the procedure until the signal samples are exhausted.

Fig. 4 illustrates the use of the above procedure for a trapezoidal pulse signal with a length of 70 samples.

1.4 Remarks

1.4.1 Advancing filters in the frequency domain

Neglecting the delay of one block inherent in FFT processing, both advancing and delaying filters can be implemented in the frequency

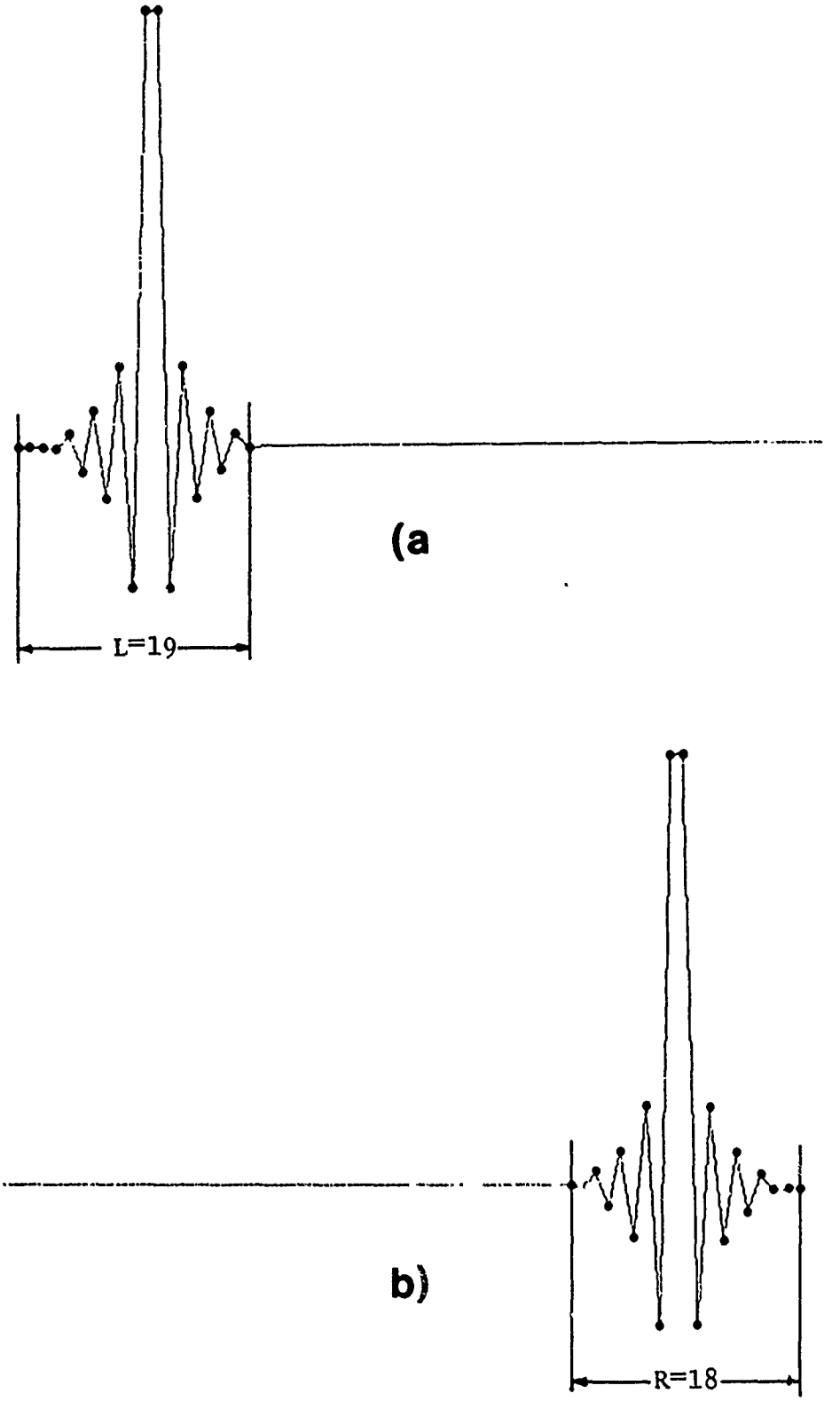


FIG. 5 IMPULSE RESPONSES OF DELAYING AND ADVANCING FILTERS

domain. Due to the cyclical nature of the DFT, advancing a signal by m samples is equivalent to delaying it by N-m samples, where N is the block size. Figure 5a shows the IR of a $10\frac{1}{2}$ smaples delaying filter of order 16 for a block size N=64. The IR of a $10\frac{1}{2}$ samples advancing filter is shown in Fig. 5b. To implement these filters by the technique described in section 1.3 one should take $L \geq 19$, $R \geq 0$ for the delaying filter, and $L \geq 0$, $R \geq 18$ for the advancing filter.

1.4.2 Trigonometric polynomial approximations

When signals are short enough to fit in the memory as a single block, and maximum speed is not required*, one can delay the signal by α samples (α may be non-integral) as follows:

1) DFT the signal.
2) Multiply by $\{\exp(-j2\pi\alpha n/N)\}$.
3) IDFT.

This is the method discussed in Refs. 2 and 3. It is easy to see that this is equivalent to interpolating the signal samples with a trigonometric polynomial [Ref. 11], and then delaying this polynomial. The procedure can also be viewed as an implementation of fast convolution with a filter IR of length equal to the block size N.** It follows from this fact that only part of the output data is to be regarded as valid (see section 1.3), unless the signal is padded with an appropriate number of zeros [Ref. 7].

1.4.3 FFT-based interpolation

Digital time-shifting is useful mainly in multichannel processing for beam steering purposes. However, it may sometimes be an

---

*For a given IR length there is a block size for which the computing time is minimum [Ref. 8].

**Note that the IDFT of $\{\exp(-j2\pi\alpha n/N)\}$ decays with $\sin x / x$ for non-integer α, and therefore the order of the filter is less than N for practical purposes.

attractive technique for interpolating a single bandlimited signal. Suppose that a bandlimited signal was sampled at the time instants $n\Delta t$, where Δt is small enough to satisfy Nyquist's condition, and that one wishes to increase the sampling rate and find the samples at $n\Delta t/K$. This problem is normally solved in practice by DFT'ing the N given signal samples, adding zeros to the DFT to obtain a block size KN, and inverting [Refs. 11 and 15]. The "expansion factor" K is chosen as a power of two for convenience in using the most common FFT programs. This procedure is equivalent to passing a trigonometric polynomial through the samples. The same result can be obtained by a slightly different computational technique. For concreteness take $K=4$. Interpolating is equivalent to delaying the signal by 1/4 of a sample, 1/2 sample, and 3/4 of a sample, and then combining the partial results (see Fig. 6). This can be done by DFT'ing the signal, multiplying by $\{\exp(-j2\pi n/KN)\}$, inverting, etc. It is shown in the Appendix that this procedure is easy to program using only 3N words of core memory, while straightforward implementation of the conventional technique requires KN words. The running time is about the same in the two cases.

A final important remark is that the time-delaying filter need not have a DFT $\{\exp(-j2\pi n/KN)\}$. Indeed, for specified frequency-domain tolerances, a filter with shorter IR will often be acceptable, and proceeding as indicated e.g. in section 1.3 will lead to faster processing and lower memory requirements.

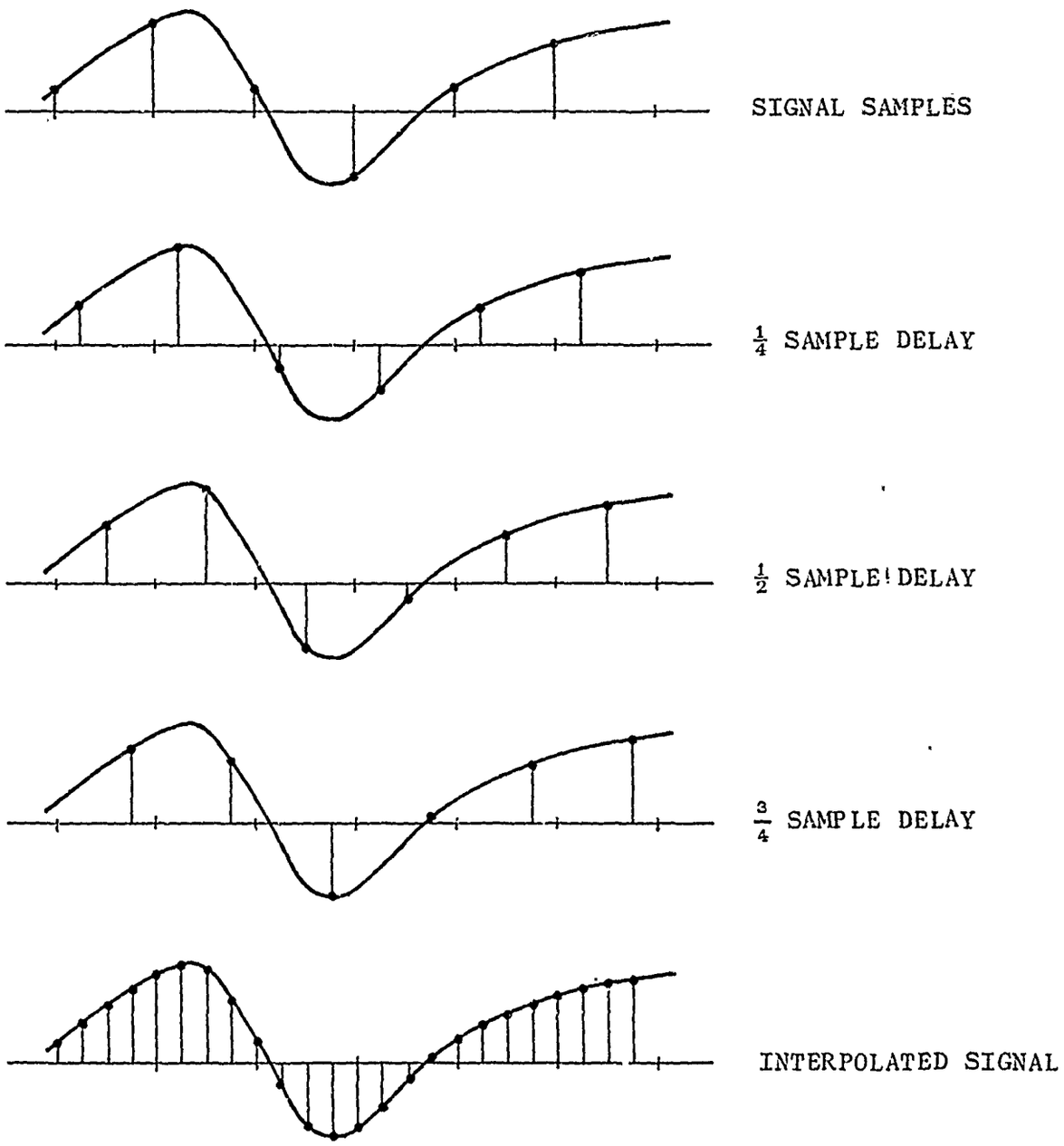


FIG. 6 INTERPOLATION BY SUCCESSIVE TIME-DELAYING

2. DIGITAL BEAMFORMING USING FIR FILTERS

2.1 General

This chapter discusses the design of digital processors for steering broadband arrays, under the assumption that some of the sensors' outputs must be shifted by non-integral number of samples. If only an integral number of samples time-shifts are needed, a straightforward shift-register type of implementation is usually to be preferred, unless complicated filtering operations besides beamforming must also be performed by the array processor. Examples of applications involving multichannel filtering include constant-beamwidth arrays [Ref. 5] and "optimal" arrays [Ref. 12]. The discussion in this chapter is relevant to this type of applications, even though no fractional delays may be needed.

Because of the large number of possible alternatives, some of which may be advantageous in particular cases, no attempts at being exhaustive will be made, and only two techniques will be described. These suffice to illustrate the principles of time-domain and frequency-domain beamforming.

An underlying assumption throughout this chapter is that the sampling frequency is the same for all sensors. It should be noted, however, that in systems with very large bandwidth such as those described in Ref. 5, wherein some sensors are effectively cut-off at high frequencies, it may be desirable to have different sampling rates.

Consider an array of M sensors placed at arbitrary locations $\{z_i\}$ on a line (Fig. 7)*, and suppose that B beams are to be formed. It will be assumed, for simplicity, that the desired beam distribution is symmetric with respect to the axis of the array

---

* The techniques described in section 2.2 below can be applied also to planar and volume arrays with only minor modifications required.

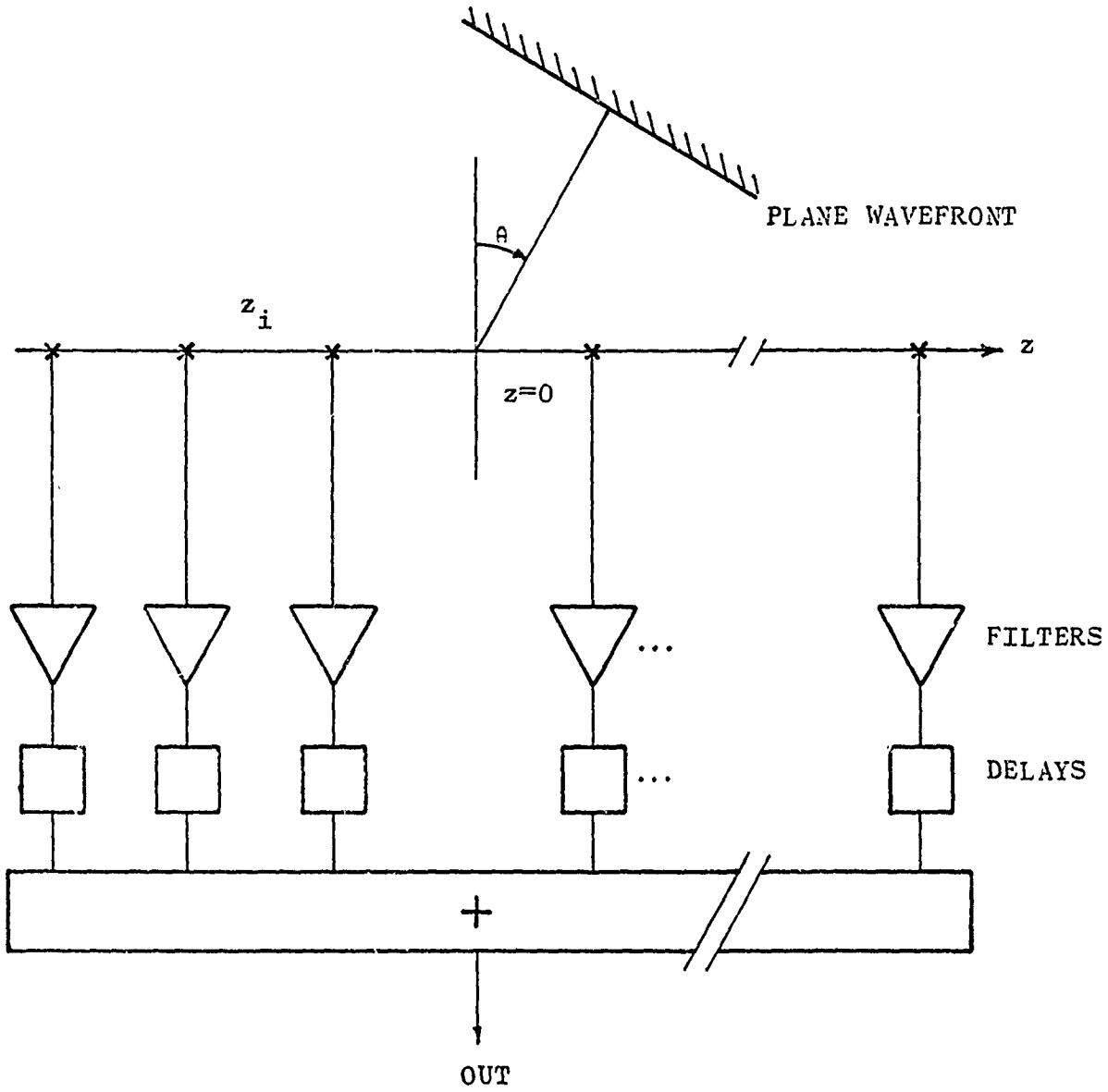


FIG. 7 ARRAY MODEL

(broadside). In a self-explanatory fashion the beams will be labeled by assigning them the numbers $-(B-1)/2, \ldots, -1, 0, 1, \ldots, (B-1)/2$. (B is assumed to be odd.) To steer the array at an angle θ from broadside one must time-shift the output of sensor $\underline{i}$ by $z_i \cdot \sin\theta / c$, where $\underline{c}$ is the sound speed. It is convenient to form the beams at equal increments of $\sin\theta$, because then it suffices to design only one time-shifting filter per sensor. For example, beam number 2 can be formed by passing the sensors' outputs twice through the filters needed for beam number 1.

The design of time-shifting filters has been treated in Chapter 1. For broadband arrays, such as those used with explosive sound sources, the linear phase characteristics must be approximated over a large portion of the available bandwidth, from close to DC to 3/4 of the Nyquist frequency, say.

The first step in the design of a beamformer consists in choosing the maximum errors admissible in the filters' amplitude and phase characteristics ($\Delta A + \Delta\phi$, respectively). This choice depends on the desired performance and will not be discussed in this paper.

The percentage P of sensors requiring delays of non-integral number of samples for the first beam can then be determined. P depends on the phase and amplitude tolerances, the sensor locations, and the beam resolution. To simplify the discussion, it will be assumed in the sequel that the same sensors need non-integer delays for all beams. Filters can be designed by the techniques outlined in Chapter 1 to introduce the time-shifts required by the PM sensors to form beam number 1. Note that the tolerances for these filters should be $\Delta\phi$ and ΔA divided by $(B-1)/2$, if the different beams are computed "recursively" as indicated above. It will be further assumed that FIR filters are used, and that the order of the filters, D, needed to achieve the specified tolerances, is the same for all filters.

Program organizations and estimates of running time and core memory requirements are considered in the following section. Two cases of interest are discussed: 1) beamforming when no prior filtering is necessary, and 2) beamforming preceded by digital filtering.

It will become apparent that the most economic solution, in terms of computing time, depends on the number of sensors, the number of beams, the percentage of sensors that require fractional delays, the tolerances (reflected on the order of the filters), and also on whether prior filtering is necessary. An example will show that frequency-domain processing is indeed a competitive technique for certain application.

2.2 Computational considerations

2.2.1 Time-domain processing with no prior filtering

A program organization, schematically indicated in Fig. 8, will now be described. Let S' be the total time-shift (in samples) between the two outermost sensors for beam $(B-1)/2$, and denote by S an integer greater or equal to S. For each sensor $S+D$ samples are kept in core. If the sensor needs no fractional delay, it suffices to scan the respective buffer, and to transfer the appropriate samples to another buffer of length B. If a fractional delay is required, for each beam D samples are taken out of the sensor buffer, convolved with an interpolating filter of length D, and the result stored in a buffer of length B. Once this is done for all sensors, one output sample per beam is obtained simply by adding the contents of the M buffers of length B. The sensor buffers are then shifted to the right by one sample, the rightmost sample is discarded, a new signal sample is introduced at the leftmost position, and the beamforming operation is again performed.

Let K_a be the real addition time, and k_m the real multiplication time for the machine being used. A rough estimate of the running time can be obtained by neglecting the time needed for input/output, addressing, and transfer of data. For each of the sensors that need fractional delays, D real multiplications and additions are required per beam and data point. Noting that forming beam zero does not involve delaying, the total time for the above operations is,

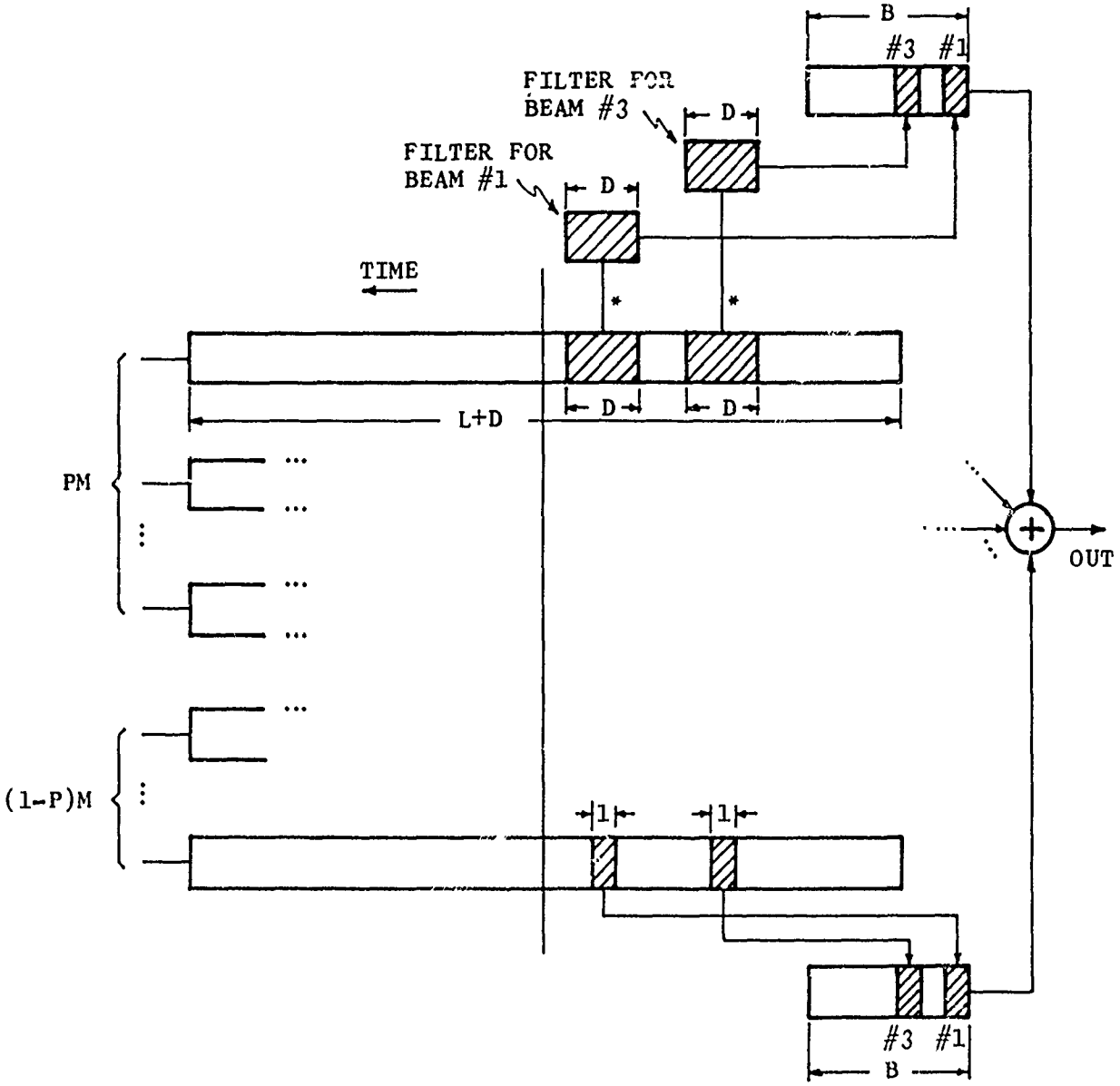


FIG. 8 TIME-DOMAIN BEAMFORMING

approximately,

$$(k_m+k_a)PMD(B-1)$$

per data point. Taking into consideration the time needed for the addition of the M buffers of length B, one obtains finally the total time per output sample*

$$T_1 = (k_m+k_a)PMD(B-1) + k_aBM$$

Assuming that an integer is represented by one computer word, the total amount of core required is

$$M(S+D) + MB + PMBD$$

words.

It was implicitly assumed that the term PMBD above is small, implying that it is easy to keep in core the various filter IR's, of length D samples each. Note that, in this assumption, each filter can be designed for the tolerances $\Delta\Phi$, ΔA.

2.2.2 Frequency-domain processing with no prior filtering

Taking the center of the array as the reference for measuring phases, beamforming involves delaying or "advancing" the outputs of the various sensors. The maximum time-shift needed is S/2 samples (S is assumed even, for simplicity). All the time-shifting filters needed can be implemented by the "select-save" technique taking L=R=(S+D)/2 (cf. section 1.3). Some of the filters could be implemented with lower values of L and R. However, the program organization is facilitated if the same values are used for all sensors.

---

* The beamformer output is to be interpreted as a vector of B components, each of which is a time-series corresponding to a beam. Thus, T_1 is the time necessary to generate B samples, one per beam.

The procedure will now be described. The DFT of the time-shifting filters needed for the first beam and each sensor are first computed and stored (on disk, say). The time required for this computation will not be taken into account, since it need only be done once for each array. For sensor number 1, N signal samples are read in, together with its filter DFT. The signal is DFT'ed and then multiplied by the filter to yield the DFT of the sensor's contribution to the first beam. This last operation takes a complex multiplication, i.e., four real multiplications and two real additions, per frequency point. Because there are only N/2+1 distinct frequency points, the time required for phase-shifting beam number 1 is

$$(4k_m+2k_a)(N/2+1)$$

per sensor and per block of data. Beam number 2 can be obtained by multiplying the DFT of beam 1 by the filter characteristics, and so forth.* Beams numbered -1, -2, etc., can be obtained by successive multiplication of the signal DFT by the complex conjugate of the delaying filter DFT. The total phase-shifting effort is

$$(4k_m+2k_a)(N/2+1)(B-1)$$

per block and per sensor. Suppose that the B frequency-domain blocks, which are the contribution of sensor 1 to the various beams can be kept in core. One can then take a block of signal 2 and its filter, and proceed as indicated for signal 1. For each beam the contribution of sensor 2 should be added to the contribution of sensor 1, and the result accumulated in core. It is easy to see that only B + 3 blocks of core are used in this

---

* By using an additional I/O operation (read a block from disk) per beam, the interpolating filters can all be designed for the tolerances ΔA, $\Delta\phi$ (cf. end of section 2.2.1 for a similar discussion in time-domain processing). This involves more initial design effort and more disk memory.

implementation. The procedure is continued for all sensors. The total time required for phase-shifting and adding is

$$(4k_m+2k_a)(N/2+1)(B-1)M + 2k_a(N/2+1)B(M-1)$$

Each beam is subsequently inverse transformed, and S+D-1 samples discarded (see section 1.3). Thus, M signal DFT's and B beam IDFT's are needed per block. Denoting by k_{fft} the computation time for a FFT with block size N, the total transforming and inverting effort per block is

$$k_{fft}(M+B)$$

Because the signals are real, k_{fft} is approximately [Ref. 8]

$$\tfrac{1}{2}(2k_m+3k_a)N\log_2 N$$

Recalling that only N-(S+D-1) points are regarded as valid data, the total computing time per output sample is therefore

$$T_2=[\tfrac{1}{2}(2k_m+3k_a)(N\log_2 N)(M+B) + (4k_m+2k_a)(N/2+1)(B-1)M + 2k_a(N/2+1)B(M-1)]/[N-(S+D-1)]$$

The block length N should be chosen so as to minimize the computing time T_2. As in the simpler case of single-channel filtering [Ref. 8], as long as N is not too small, the running time is not strongly dependent on the block length (see section 2.2.5 below).

The time necessary to perform the I/O of 2M+B blocks of N samples was neglected in the above estimates.

The procedure described uses (B+3)N words of core for its data manipulations (of course other arrangements are possible). If no external storage device such as a disk is available, frequency-domain processing takes a prohibitive amount of core and will not be feasible unless a very large computer is used. Because of the large number of variables involved, an attempt at

comparing time- and frequency-domain processing in general terms will not be made. However, a discussion based on a specific example will be given in section 2.2.5 below.

2.2.3 Time-domain processing with prior filtering

Suppose now that the signals must be filtered prior to beamforming by means of FIR filters with IR's of length Q samples.

The time required to implement the filtering in the time domain by direct convolution is

$$(k_m+k_a)Q$$

per sensor and data point. Using the results and assumptions of section 2.2.1, the total computing time for filtering and beamforming is

$$T_3=(k_m+k_a)QM + (k_m+k_a)PMD(B-1) + k_aBM$$

Since additional buffers are needed for the filtering, the memory requirements are increased by 2MQ words.

2.2.4 Frequency-domain processing with prior filtering

The procedure is similar to that described in section 2.2.2, except that one should take $L=R=Q+S+D$ for the "select-save" technique, and an additional complex block multiplication, together with an I/O operation are needed per sensor. Thus, each block requires an additional

$$(4k_m+2k_a)(N/2+1)M$$

The total time per output sample is now

$$T_4=[\tfrac{1}{2}(2k_m+3k_a)(N\ \log_2 N)(M+B) + (4k_m+2k_a)(N/2+1)MB + 2k_a(N/2+1)B(M-1)]/[N-(Q+S+D-1)]$$

N should be chosen by minimizing the above expression. Note that the optimal value of N is now also dependent on the order of the filters Q.

The core memory requirements remain the same as in section 2.2.2.

It is apparent that the running time is only slightly increased by the filtering operation when the processing is done in the frequency domain. In time-domain processing, however, the increase may be substantial.

2.2.5 Example

The computational considerations of the preceding sections will now be applied to a specific array, which is presently being studied at SACLANTCEN [Ref. 5].

The array has 20 unequally spaced hydrophones, the beam-width is approximately constant at about 15° over the $3\frac{1}{2}$ octave band of operation, and it is desired to form 5 beams at -30°, -15°, 0°, 15°, and 30°.

For a sampling frequency F_s and array length d, the quantity S' defined in section 2.2 is given by

$$S' = \sin \theta_M . F_s . d/c$$

where c is the sound speed, and θ_M is the largest steering angle desired. For the example under consideration $S' \simeq 84$. To achieve constant beamwidth [Ref. 5] it is necessary to filter the sensor outputs by means of FIR filters of order Q=256. The outer hydrophones in the array are effectively cut-off at high frequency and do not need fractional delays. However, the 8 central hydrophones require non-integral time-shifts, whence P=8/20. The interpolating filters for the central sensors are of order D=16. The relevant parameters for the array are

therefore:

$$M = 20$$
$$B = 5$$
$$P = 8/20$$
$$D = 16$$
$$S = 84$$
$$Q = 256$$

The processor is implemented in a Hewlett-Packard 2116B minicomputer with disk storage and extended arithmetic unit, whose multiply and add times are*, in microseconds,

$$k_m = 25.6$$
$$k_a = 9.6$$

The running time for time-domain processing, found by direct substitution in the expressions of section 2.2.3, is

$$T_3 \simeq 180 + 18 + 1 \simeq 200 \text{ msec/sample.}$$

The largest contribution to T_3 comes from the filtering operation.

For frequency-domain processing N must be chosen so as to make T_4 small (see section 2.2.4). Substitution into the expression found in section 2.2.4 yields (in microseconds)

$$T_4 \simeq (1000 \log_2 N + 6080 + 810).N/(N-355).$$

T_4 varies slowly with N, when N is above a "threshold", and

---

* These figures include fetching a word, fixed-point multiplying or adding, and storing the result, and are therefore somewhat pessimistic. On the other hand, the computing effort involved in checking and correcting for overflows is not taken into consideration in the estimates.

therefore the choice of block size is not critical. (A similar result obtains for single-channel filtering [Ref. 8].) Table I shows values of T_4 corresponding to various block sizes (powers of two).

TABLE I

N	T_4 (msec/sample)
512	49
1024	26
2048	22
4096	21

It is apparent that the running time is about the same as long as the block size is larger than 512. Thus, 1024 is a good choice for N, since the amount of core memory needed is smaller. Still, about 8k words of core memory are used in this implementation, while time-domain processing only requires in the order of 3k. Frequency-domain processing is about 8 times faster than time-domain processing for this particular example. This is not so surprising, since a large effort must be put into the filtering operation. If the order of the filters were lower and the number of beams to be formed higher the economics of the process might change.

Consider now the same example, but assume that no prior filtering is required. The computing time for time-domain processing is now

$$T_1 \simeq 19 \text{ msec/sample},$$

and the memory needed under 1k. For frequency-domain processing the choice of block length must be reconsidered. The computing time now becomes

$$T_2 \simeq (1000 \log_2 N + 4860 + 810) \cdot N/(N-100).$$

T_2 is given in Table II for various block sizes.

TABLE II

N	T_2 (msec/sample)
256	22
512	18
1024	17.5
2048	17.5

Somewhat unexpectedly, the computing time for FFT processing is less than for time-domain processing in this example, even if no filtering is needed. This fact should not be taken as a general result, but rather as evidence that FFT processing can indeed be a competitive technique for beamsteering.

A final comment is warranted. For a sampling frequency $F = 24$ kHz the array described above will take in the order of 6 minutes to process a signal of length 1 second. Since the operation can be accelerated by using a faster computer, a hardware FFT transformer and careful programming, the use of broadband arrays for explosive echo-ranging seems possible with state-of-the-art techniques. It should be noted that the computing time estimates given throughout this chapter are very rough and should be considered as order-of-magnitude values. The ratio between running time for frequency domain and time-domain processing is likely to be more accurate than each of the separate estimates. Measured values for the running time of the overall processors are not yet available. However the time for computing a 1024 point FFT using routines supplied by Hewlett-Packard has been found to be about three times longer than the values used in the estimates of this chapter. The discrepancy is believed to be due to book-keeping and scaling operations needed by the fixed-point routines. Errors of the same order of magnitude may well be present in all the given estimates.

RECAPITULATION & CONCLUSIONS

Digital filtering theory provides convenient and powerful means of treating broadband digital beamforming, especially when time-shifts of non-integral numbers of samples are needed. This problem arises when the sampling frequency is too low for accurate beamsteering to be possible by using only time-shifts of an integral number of samples. Keeping the sampling frequency as low as possible is necessary when minimal data acquisition rates are desired* as often happens in experiments with hydrophone arrays and explosive sound sources. The digital filtering approach is also advantageous when the sensors' outputs must be filtered prior to beamforming, especially if the filtering operation is fairly complex, as e.g. in constant beamwidth arrays [Ref. 5], and "optimal" arrays [Ref. 12].

Time-shifting (interpolating) filters form the essential part of the steering processor. FIR filters are particularly attractive as interpolators, because they are quite simple to design given phase and amplitude tolerances in the frequency domain.** Implementation of these filters is also straightforward, and no restriction on signal duration is necessary. It is shown in Chapter 1 that high accuracy can be obtained with low order FIR filters without recourse to sophisticated design techniques.

Beamforming procedures based on time-domain and frequency-domain implementation of FIR filters are discussed in Chapter 2 and rough estimates of running time and core-memory requirements are given.

---

* A trade-off exists between the complexity of the data acquisition equipment to achieve a certain data rate and the complexity of the processing. If the data acquisition system and the processor are being designed simultaneously, this trade-off should be taken into consideration in the choice of the sampling frequency.

** How to specify these tolerances depends on the particular problem being considered, and is not discussed in this paper.

The estimates depend on the number of sensors, the number of beams, the percentage of sensors needing fractional delays, and the prescribed accuracy. For each particular case these procedures can be compared by simple substitution of the relevant parameters in the formulae derived. Refinements of these techniques, as well as "hybrid" processing (partly in the time domain and partly in the frequency domain) are easy to evaluate along the same lines.

Sample calculations performed for an array of interest in SACLANTCEN's work involving explosive sound sources indicate that frequency-domain processing can be competitive, in terms of computing time, for beamforming alone and may give 1:10 savings when complicated prior filtering is needed. Frequency-domain methods are disadvantageous from the memory use viewpoint. The order of magnitude of the running time for the example considered shows that directional arrays using in the order of 20 or 50 hydrophones for broadband echo-ranging are within present day's digital processing technology.

The problem of estimating the power spectrum of a signal using an array [Ref. 3] is not discussed in this paper. Suffice it to say, however, that the array can be steered and the output obtained to any prescribed frequency-domain accuracy by the techniques described. From the array output the power spectrum can then be estimated by standard techniques (see e.g. Refs. 13 and 14).

APPENDIX

INTERPOLATION USING THE FFT

Estimates for computing time and core use are derived in this appendix for the two FFT interpolation techniques described in section 1.4.3.

1. DFT-add zeros-IDFT

The computing time is, using the same notation as in section 2.2,

$$T=\tfrac{1}{2}(2k_m+3k_a)\cdot[N \log_2 N + KN \log_2(KN)],$$

and KN words of core are needed for a straightforward implementation.

2. Successive delaying

The procedure is as follows. First compute and store in core the phase-shifting filter $\{\exp(-j2\pi n/KN)\}$. Then, DFT the signal, phase-shift, store, and invert. The IDFT can be written onto disk, say, if the amount of available core memory is small. A second phase-shift and inversion can then be done, and so forth. K-1 phase-shifts are necessary to obtain all the interpolated values. Hence, the running time is

$$T'=\tfrac{1}{2}(2k_m+3k_a)KN\cdot\log_2 N + (4k_m+2k_a)(K-1)(N/2+1)$$

if the time to compute the filter characteristics is neglected. The amount of core memory needed, 3N words, is less than in the previous method. Of course some unscrambling must be done in this procedure, to sort the samples in the correct order. The time required for this operation is usually small and is not taken into account in the assumate above.

Assuming that $k_a << k_m$, $K << N$, and $\log_2 N >> 2$, it is easy to see that $T \approx T'$. Therefore the two techniques lead to about the same computing time. When the expansion factor K is large, a recently proposed modification of the FFT algorithm (Ref. 17) may lead to considerable time-savings and make the "DFT-add zeros-IDFT" technique more attractive. Succesive delaying is advantageous from a memory requirements point of view. It should be noted, however, that it is possible to compute a FFT of block size KN, without using KN words of core (Ref. 16).

The above discussion shows, incidentally, that a straightforward IDFT interpolation technique is inferior to "DFT-phase shift-IDFT" for beamsteering purposes, both in running time and core use. In fact, it would only be about as good if all the KN samples generated were needed in the beamforming algorithm.

REFERENCES

1. A. Barbagelata, A. Castanet, R. Laval and M. Pazzini, "A High-Density Digital Recording System for Underwater Sound Studies", SACLANTCEN Technical Report No. 170, July 1970, NATO UNCLASSIFIED.

2. J.R. Williams, "Fast Beamforming Algorithm", J.A.S.A., Vol. 44, No. 5, pp. 1454-1455, November 1968.

3. P. Rudnick, "Digital Beamforming in the Frequency Domain", J.A.S.A., Vol. 46, No. 5, pp. 1089-1090, November 1969.

4. L.R. Rabiner, "Techniques for Designing Finite-Duration Impulse-Response Digital Filters", IEEE Trans. on Comm. Tech., Vol. COM-19, No. 2, pp. 188-195, April 1971.

5. A.A.G. Requicha, "Design of Wideband Constant-Beamwidth Acoustic Arrays", SACLANTCEN Technical Report No. 205, January 1971, NATO UNCLASSIFIED.

6. T.G. Stockham, Jr., "High Speed Convolution and Correlation", AFIPS Conf. Proc., Vol. 28, Spring Joint Computer Conf., pp. 229-233, Washington: Spartan Books, 1966.

7. T.G. Stockham, Jr., "High Speed Convolution and Correlation with Applications to Digital Filtering", in "Digital Processing of Signals", B. Gold and C.R. Rader, eds., New York: McGraw-Hill, 1969.

8. H.D. Helms, "Fast Fourier Transform Method of Computing Difference Equations and Simulating Filters", IEEE Trans. on Audio and Electroac., Vol. AU-15, pp. 85-90, June 1967.

9. H.D. Helms, "Digital Filters with Equiripple or Minimax Responses", IEEE Trans. on Audio and Electroac., Vol. AU-19, pp. 87-92, March 1971.

10. B. Gold and C.R. Rader, "Digital Processing of Signals", New York: McGraw-Hill, 1969.

11. W.M. Gentleman and G. Sande, "Fast Fourier Transforms-for Fun and Profit", AFIPS Conf. Proc., Vol. 29, Fall Joint Computer Conf., pp. 563-578, Washington: Spartan Books, 1966.

12. C.W. Horton, Sr., Signal Processing of Underwater Acoustic Waves, Chapter 12, Washington: U.S. Government Printing Office, 1969.

13. J.W. Cooley, P.A.W. Lewis, P.D. Welch, "The Application of the Fast Fourier Transform Algorithm to the Estimation of Spectra and Cross-Spectra", J. Sound Vib., Vol. 12, No. 13, pp. 339-352, 1970.

14. L.D. Enochson and R.K. Otnes, Programming and Analysis for Digital Time Series Data, Washington: Navy Publication and Printing Service Office, 1968.

15. J.M. Howem, "(sin x)/x Interpolation of Sampled Signals", SACLANTCEN Technical Report No. 196, July 1971, NATO UNCLASSIFIED.

16. N.M. Brenner, "Fast Fourier Transform of Externally Stored Data", IEEE Trans. on Audio and Electroac., Vol. AU-17, No. 2, pp. 128-132, June 1969.

17. J.D. Markel, "FFT Pruning", IEEE Trans. on Audio and Electroac., Vol. AU-19, No. 4, pp. 305-311, Dec. 1971.